

IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD

Publication number: JP2006518056 (T)

Publication date: 2006-06-03

Inventor(s):

Applicant(s):

Classification:






- International: G06F11/00; G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78; G06F11/00; G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/76

- European: G06F9/3854

Application number: JP20050502223T 20031031

Priority number(s): US20030422503P 20021031; US20030683929 20031009; US20030683932 20031009; US20030684053 20031009; US20030684057 20031009; US20030684102 20031009; WO2003US34556 20031031

Also published as:

 WO2004042560 (A2) WO2004042560 (A3) WO2004042574 (A2) WO2004042574 (A3) WO2004042569 (A2)

more >>

Abstract not available for JP 2006518056 (T)

Abstract of corresponding document: **WO 2004042560 (A2)**

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations. By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.

Data supplied from the **esp@cenet** database — Worldwide

(19) 日本国特許庁(JP)

(12) 公表特許公報(A)

(11) 特許出願公表番号

特表2006-518056

(P2006-518056A)

(43) 公表日 平成18年8月3日(2006.8.3)

(51) Int. Cl.
G06F 11/00 (2006.01)F I
G06F 9/06 630Aテーマコード (参考)
SB076
SB176

審査請求 未請求 予備審査請求 未請求 (全 27 頁)

(21) 出願番号 特願2005-502223 (P2005-502223)
 (86) (22) 出願日 平成15年10月31日 (2003.10.31)
 (85) 翻訳文提出日 平成17年6月16日 (2005.6.16)
 (86) 国際出願番号 PCT/US2003/034556
 (87) 国際公開番号 W02004/042569
 (87) 国際公開日 平成16年5月21日 (2004.5.21)
 (31) 優先権主張番号 60/422,503
 (32) 優先日 平成14年10月31日 (2002.10.31)
 (33) 優先権主張国 米国 (US)
 (31) 優先権主張番号 10/683,929
 (32) 優先日 平成15年10月9日 (2003.10.9)
 (33) 優先権主張国 米国 (US)
 (31) 優先権主張番号 10/683,932
 (32) 優先日 平成15年10月9日 (2003.10.9)
 (33) 優先権主張国 米国 (US)

(71) 出願人 504242618
 ロッキード マーティン コーポレーション
 アメリカ合衆国 メリーランド州 208
 17-1803 ベセスダ ロックレッヂ
 ドライブ 6801
 (74) 代理人 100083932
 弁理士 廣江 武典
 (74) 代理人 100129698
 弁理士 武川 隆寛
 (74) 代理人 100129676
 弁理士 ▲高▼荒 新一
 (74) 代理人 100130074
 弁理士 中村 繁元

最終頁に続く

(54) 【発明の名称】 プログラマブル回路、関連計算マシン、並びに、方法

(57) 【要約】

プログラマブル回路は、外部ソースからコンフィギュレーション・データを受信し、ファームウェアをメモリに記憶してから、そのメモリからそのファームウェアをダウンロードする。そうしたプログラマブル回路は、計算マシン等のシステムにプログラマブル回路のコンフィギュレーションを変更させて、コンフィギュレーション・メモリをマニュアルで再プログラムする必要性をなくしている。例えば、もしプログラマブル回路がパイプライン加速器の部分であるFPGAであれば、その加速器と結合されたプロセッサはFPGAのコンフィギュレーションを変更できる。より詳細には、プロセッサはコンフィギュレーション・レジスタから変更されたコンフィギュレーションを表すファームウェアを検索し、そのファームウェアをFPGAに送信し、次いで該FPGAがそのファームウェアを電氣的に消去可能でプログラム可能な読み取り専用メモリ (EEPROM) 等のメモリに記憶する。次に、FPGAはメモリからそのコンフィギュレーション・レジスタにそのファームウェアをダウンロードし、よってそれ自体を変更されたコンフィギュレーションを有するように再構成する。

10

【特許請求の範囲】

【請求項 1】

プログラマブル回路であって、
外部ソースからコンフィギュレーションを表すファームウェアを受信し、
前記ファームウェアをメモリに記憶し、
前記メモリから前記ファームウェアをダウンロードするように動作できるプログラマブル回路。

【請求項 2】

前記ファームウェアの前記メモリからのダウンロード後に前記コンフィギュレーションで更に動作できる、請求項 1 に記載のプログラマブル回路。

10

【請求項 3】

前記メモリが不揮発性メモリを含む、請求項 1 に記載のプログラマブル回路。

【請求項 4】

前記メモリが外部メモリを含む、請求項 1 に記載のプログラマブル回路。

【請求項 5】

プログラマブル回路であって、
メモリから第 1 コンフィギュレーションを表す第 1 ファームウェアをダウンロードし、
前記第 1 コンフィギュレーションで動作し、
前記メモリから第 2 コンフィギュレーションを表す第 2 ファームウェアをダウンロードし、
前記第 2 コンフィギュレーションで動作できるプログラマブル回路。

20

【請求項 6】

前記プログラマブル回路が、
前記第 1 コンフィギュレーションで動作している一方で外部ソースから前記第 2 ファームウェアを受信し、
前記第 1 コンフィギュレーションで動作している一方で前記メモリに前記第 2 ファームウェアを記憶するように更に動作できる、請求項 5 に記載のプログラマブル回路。

【請求項 7】

プログラマブル回路ユニットであって、
メモリと、
前記メモリと結合されたプログラマブル回路と、を含み、
前記プログラマブル回路が、
前記プログラマブル回路のコンフィギュレーションを表すファームウェアを外部ソースから受信し、
前記ファームウェアを前記メモリに記憶し、
前記メモリから前記ファームウェアをダウンロードするように動作できるプログラマブル回路と、
を含むことから成るプログラマブル回路ユニット。

30

【請求項 8】

前記メモリが電氣的に消去可能でプログラム可能な読み取り専用メモリを含む、請求項 7 に記載のプログラマブル回路ユニット。

40

【請求項 9】

前記プログラマブル回路がフィールド・プログラマブル・ゲート・アレイを含む、請求項 7 に記載のプログラマブル回路ユニット。

【請求項 10】

プログラマブル回路ユニットであって、
それぞれ第 1 及び第 2 のコンフィギュレーションを表す第 1 及び第 2 のファームウェアを記憶するように動作できるメモリと、
前記メモリと結合されたプログラマブル回路と、を含み、
前記プログラマブル回路が、

50

前記メモリから前記第1ファームウェアをダウンロードし、

前記第1コンフィギュレーションで動作し、

前記メモリから第2ファームウェアをダウンロードし、

前記第2コンフィギュレーションで動作するように動作できことから成るプログラマブル回路ユニット。

【請求項11】

前記プログラマブル回路が、

前記第1コンフィギュレーションで動作している一方で外部ソースから前記第2ファームウェアを受信し、

前記第1コンフィギュレーションで動作している一方で前記メモリに前記第2ファームウェアを記憶するように更に動作できる、請求項10に記載のプログラマブル回路ユニット。

10

【請求項12】

前記プログラマブル回路が、前記第1コンフィギュレーションで動作している一方で前記第2ファームウェアをロードするように動作できる、請求項10に記載のプログラマブル回路ユニット。

【請求項13】

プログラマブル回路ユニットであって、

それぞれ、第1、第2、第3、並びに、第4のコンフィギュレーションを表す第1、第2、第3、並びに、第4のファームウェアを記憶するように動作できるメモリと、

20

前記メモリと結合された第1プログラマブル回路と、を含み、

前記第1プログラマブル回路が、

前記メモリから前記第1ファームウェアをダウンロードし、

前記第1コンフィギュレーションで動作し、

前記メモリから第2ファームウェアをダウンロードし、

前記第2コンフィギュレーションで動作するように動作でき、

前記メモリと結合されると共に前記第1プログラマブル回路と結合された第2プログラマブル回路を更に含み、

前記第2プログラマブル回路が、

前記メモリから前記第3ファームウェアをダウンロードし、

30

前記第3コンフィギュレーションで動作し、

前記メモリから第4ファームウェアをダウンロードし、

前記第4コンフィギュレーションで動作するように動作できることから成るプログラマブル回路ユニット。

【請求項14】

前記第1プログラマブル回路が、

前記第1コンフィギュレーションで動作している一方で外部ソースから前記第2及び第4のファームウェアを受信し、

前記第1コンフィギュレーションで動作している一方で前記メモリに前記第2及び第4のファームウェアを記憶するように動作できる、請求項13に記載のプログラマブル回路ユニット。

40

【請求項15】

前記第1及び第2のプログラマブル回路が各フィールド・プログラマブル・ゲート・アレイを含む、請求項13に記載のプログラマブル回路ユニット。

【請求項16】

計算マシンであって、

プロセッサと、

前記プロセッサと結合されたプログラマブル回路ユニットと、を含み、

前記プログラマブル回路ユニットが、

メモリと、

50

前記メモリと結合されたプログラマブル回路と、を含み、

前記プログラマブル回路が、

前記プログラマブル回路のコンフィギュレーションを表すファームウェアを前記プロセッサから受信し、

前記メモリに前記ファームウェアを記憶し、

前記プロセッサに応じて前記メモリから前記ファームウェアをダウンロードするように動作できることから成る計算マシン。

【請求項 17】

前記プロセッサが、

前記ファームウェアを前記プログラマブル回路に送信する前、そのファームウェアが前記メモリに既に記憶されているかを決定し、

前記ファームウェアが前記メモリに未だ記憶されていない場合だけ、前記プログラマブル回路に前記ファームウェアを送信するように動作できる、請求項 16 に記載の計算マシン。

【請求項 18】

前記プロセッサと結合されると共に、前記ファームウェアを記憶するように且つ前記ファームウェアが前記プログラマブル回路用の所望コンフィギュレーションを表していることを示すように動作できるコンフィギュレーション・レジストリと、

前記プロセッサが前記ファームウェアを前記コンフィギュレーション・レジストリから前記プログラマブル回路にダウンロードするように動作できることと、
を更に含む、請求項 16 に記載の計算マシン。

【請求項 19】

前記プログラマブル回路ユニットがパイプライン・ユニットを含み、

前記プログラマブル回路がデータに対して動作するように動作できるハードウェアに組み込まれたパイプラインを含む、請求項 16 に記載の計算マシン。

【請求項 20】

計算マシンであって、

プロセッサと、

前記プロセッサと結合されたプログラマブル回路ユニットと、を含み、

前記プログラマブル回路ユニットが、

それぞれ第 1 及び第 2 のコンフィギュレーションを表す第 1 及び第 2 のファームウェアを記憶するように動作できるメモリと、

プログラマブル回路と、を含み、

前記プログラマブル回路が、

前記メモリから前記第 1 ファームウェアをダウンロードし、

前記第 1 コンフィギュレーションで動作し、

前記プロセッサに応じて前記メモリから前記第 2 ファームウェアをダウンロードし、

前記第 2 コンフィギュレーションで動作するように、

動作できることから成る計算マシン。

【請求項 21】

前記プロセッサが第 1 試験ポートを含み、

前記プログラマブル回路ユニットが前記第 1 試験ポートと結合された第 2 試験ポートを含み、

前記プロセッサが前記第 1 及び第 2 の試験ポートを介して前記第 1 ファームウェアをメモリにロードするように動作できる、請求項 20 に記載の計算マシン。

【請求項 22】

前記プロセッサが第 1 試験ポートを含み、

前記プログラマブル回路ユニットが前記第 1 試験ポートと結合された第 2 試験ポートを含み、

前記第1コンフィギュレーションで動作している一方で、前記プログラマブル回路が自己試験を実行するように且つ自己試験データを前記第1及び第2の試験ポートを介して前記プロセッサに提供するように動作でき、

前記プロセッサが、前記自己試験データが前記自己試験の所定結果を示す場合だけ、前記プログラマブル回路に前記メモリから前記第2ファームウェアをダウンロードさせるように動作できる、請求項20に記載の計算マシン。

【請求項23】

前記プロセッサが前記第2ファームウェアを前記プログラマブル回路に送信するように動作でき、

前記第1コンフィギュレーションで動作している一方で、前記プログラマブル回路が前記プロセッサに応じて前記第2ファームウェアを前記メモリにロードするように動作できる、請求項20に記載の計算マシン。

【請求項24】

計算マシンであって、

プロセッサと、

前記プロセッサと結合されたプログラマブル回路ユニットと、を含み、

前記プログラマブル回路ユニットが、

それぞれ第1、第2、第3、並びに、第4のコンフィギュレーションを表す第1、第2、第3、並びに、第4のファームウェアを記憶するように動作できるメモリと、

前記メモリと結合された第1プログラマブル回路であり、

前記メモリから前記第1ファームウェアをダウンロードし、

前記第1コンフィギュレーションで動作し、

前記プロセッサに応じて前記メモリから第2ファームウェアをダウンロードし、

前記第2コンフィギュレーションで動作するように、

動作できることから成る第1プログラマブル回路と、

前記メモリと結合されると共に前記第1プログラマブル回路と結合された第2プログラマブル回路であり、

前記メモリから前記第3ファームウェアをダウンロードし、

前記第3コンフィギュレーションで動作し、

前記プロセッサに応じて前記メモリから第4ファームウェアをダウンロードし、

前記第4コンフィギュレーションで動作するように、

動作できる第2プログラマブル回路と、

を含む、計算マシン。

【請求項25】

前記プロセッサが第1試験ポートを含み、

前記プログラマブル回路ユニットが前記第1試験ポートと結合された第2試験ポートを含み、

前記プロセッサが前記第1及び第2の試験ポートを介して前記第1及び第2のファームウェアをメモリにロードするように動作できる、請求項24に記載の計算マシン。

【請求項26】

前記プロセッサが第1試験ポートを含み、

前記プログラマブル回路ユニットが前記第1試験ポートと結合された第2試験ポートを含み、

前記第1コンフィギュレーションで動作している一方で、前記第1プログラマブル回路が第1自己試験を実行するように且つ第1自己試験データを前記第1及び第2の試験ポートを介して前記プロセッサに提供するように動作でき、

前記第3コンフィギュレーションで動作している一方で、前記第2プログラマブル回路が第2自己試験を実行するように且つ第2自己試験データを前記第1及び第2の試験ポートを介して前記プロセッサに提供するように動作でき、

前記プロセッサが、前記第1及び第2のプログラマブル回路に、前記第1及び第2の自

10

20

30

40

50

已試験データが前記第 1 及び第 2 の自己試験の各所定結果を示す場合だけ、前記メモリから第 2 及び第 4 のファームウェアをそれぞれロードさせるように動作できる、請求項 2 4 に記載の計算マシン。

【請求項 2 7】

前記プロセッサが前記第 2 及び第 4 のファームウェアを前記第 1 プログラマブル回路に送信するように動作でき、

前記第 1 コンフィギュレーションで動作している一方で、前記第 1 プログラマブル回路が前記プロセッサに応じて前記第 2 及び第 4 のファームウェアを前記メモリにロードするように動作できる、請求項 2 4 に記載の計算マシン。

【請求項 2 8】

10

前記メモリが、

前記第 1 プログラマブル回路と結合され且つ前記第 1 及び第 2 のファームウェアを記憶するように動作できる第 1 メモリ区分と、

前記第 1 及び第 2 のプログラマブル回路と結合され且つ前記第 3 及び第 4 のファームウェアを記憶するように動作できる第 2 メモリ区分と、

を含む、請求項 2 4 に記載の計算マシン。

【請求項 2 9】

前記第 1 及び第 2 のメモリ区分が第 1 及び第 2 の集積回路上にそれぞれ配置されている、請求項 2 8 に記載の計算マシン。

【請求項 3 0】

20

方法であって、

プログラマブル回路にその回路のコンフィギュレーションを表すファームウェアを提供し、

前記プログラマブル回路によって前記コンフィギュレーション・データをメモリに記憶し、

前記メモリから前記プログラマブル回路に前記コンフィギュレーション・データをダウンロードすることを含む方法。

【請求項 3 1】

前記メモリから前記コンフィギュレーション・データをダウンロードした後、前記コンフィギュレーションで動作すること、

30

を更に含む、請求項 3 0 に記載の方法。

【請求項 3 2】

方法であって、

第 1 コンフィギュレーションを表す第 1 ファームウェアをプログラマブル回路にダウンロードし、

前記プログラマブル回路を前記第 1 コンフィギュレーションで動作し、

第 2 コンフィギュレーションを表す第 2 ファームウェアを前記プログラマブル回路にダウンロードし、

前記第 2 ファームウェアをダウンロードした後、前記プログラマブル回路を前記第 2 コンフィギュレーションで動作することを含む方法。

40

【請求項 3 3】

前記第 2 ファームウェアをダウンロードすることが、

前記第 2 ファームウェアを前記プログラマブル回路に送信し、

前記プログラマブル回路が前記第 1 コンフィギュレーションで動作している一方で、該プログラマブル回路によって前記第 2 ファームウェアをメモリにロードし、

前記メモリから前記プログラマブル回路に前記第 2 ファームウェアをダウンロードすることを含む、請求項 3 2 に記載の方法。

【請求項 3 4】

前記第 2 ファームウェアをダウンロードすることが、

前記第 2 ファームウェアが前記プログラマブル回路と結合されたメモリに記憶されてい

50

るかを決定し、

前記第2ファームウェアが前記メモリに記憶されていない場合だけ、前記第2ファームウェアを前記プログラマブル回路に送信し、

前記プログラマブル回路が前記第1コンフィギュレーションで動作している一方で、該プログラマブル回路によって前記第2ファームウェアを前記メモリにロードし、

前記メモリから前記プログラマブル回路に前記ファームウェアをダウンロードすることを含む、請求項32に記載の方法。

【請求項35】

前記第1コンフィギュレーションで前記プログラマブル回路を動作することが、該プログラマブル回路を試験することを含み、

前記第2ファームウェアをダウンロードすることが前記プログラマブル回路が前記試験を通過した場合だけ前記第2ファームウェアをダウンロードすることを含む、請求項32に記載の方法。

【請求項36】

方法であって、

第1及び第2のファームウェアを第1及び第2のプログラマブル回路にそれぞれダウンロードし、

前記第1及び第2のプログラマブル回路を前記第1及び第2のコンフィギュレーションでそれぞれ動作し、

第3及び第4のファームウェアを前記第1及び第2のプログラマブル回路に、前記第1プログラマブル回路を介して、それぞれダウンロードし、

前記第1及び第2のプログラマブル回路を前記第3及び第4のコンフィギュレーションでそれぞれ動作することを含む方法。

【請求項37】

前記第1及び第2のファームウェアをダウンロードすることが、前記第1及び第2のファームウェアを前記第1及び第2のプログラマブル回路に、試験ポートを介して、ダウンロードすることを含む、請求項36に記載の方法。

【請求項38】

前記第1及び第2のプログラマブル回路を前記第1及び第2のコンフィギュレーションで動作することが、前記第1及び第2のプログラマブル回路を試験することを含み、

前記第3及び第4のファームウェアを前記第1及び第2のプログラマブル回路にロードすることが、

前記試験が、前記第1プログラマブル回路が所望通りに機能していることを示す場合だけ、前記第3ファームウェアをロードし、

前記試験が、前記第2プログラマブル回路が所望通りに機能していることを示す場合だけ、前記第4ファームウェアをロードすることを含む、請求項36に記載の方法。

【発明の詳細な説明】

【技術分野】

【0001】

＜優先権の請求＞

この出願は、下記の特許文献1に対する優先権を請求するものであり、引用することによってここに合体させる。

【特許文献1】 米国仮出願第60/422,503号（2002年10月31日出願）

【0002】

＜関連出願の相互参照＞

この出願は、「改善された計算アーキテクチャ、関連システム、並びに、方法」と題された下記の特許文献2、「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された下記の特許文献3、「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された下記の特許文献4、「多数パイプライン・ユニットを有するパイプライン加速器、関連計算マシン、並びに、方法

」と題された下記の特許文献5と関連し、これら特許文献は全て2003年10月9日に出力され、其の所有者を有し、引用することでここに合体させる。

【特許文献2】米国出願第10/684,102号

【特許文献3】米国出願第10/684,053号

【特許文献4】米国出願第10/683,929号

【特許文献5】米国出願第10/683,932号

【背景技術】

【0003】

比較的大量のデータを比較的小さい期間で処理する通常の計算アーキテクチャは、処理負担を分担する多数の相互接続プロセッサを含む。処理負担を分担することによって、これら多数のプロセッサは、しばしば、所与のクロック周波数で単一プロセッサができるものよりよりも迅速にデータを処理できる。例えば、これらプロセッサの各々はデータの各部分を処理できるか、或は、処理アルゴリズムの各部分を実行できる。

【0004】

図1は、多数プロセッサ・アーキテクチャを有する従来の計算マシン10の概略ブロック図である。この計算マシン10は、マスター・プロセッサ12と、相互に通信すると共に該マスター・プロセッサとバス16を介して通信する共同プロセッサ14₁〜14_nと、遠隔装置（図1では不図示）から生データを受け取る入力ポート18と、該遠隔装置に処理データを提供する出力ポート20とを含む。また、計算マシン10はマスター・プロセッサ12に対するメモリ22と、共同プロセッサ14₁〜14_nに対する各メモリ24₁〜24_nと、マスター・プロセッサ及び共同プロセッサがバス16を介して共有するメモリ26とを含む。メモリ22はマスター・プロセッサ12に対するプログラム及び作業メモリの双方の役割を果たし、各メモリ24₁〜24_nは各共同メモリ14₁〜14_nに対するプログラム及び作業メモリの双方の役割を果たす。共有されたメモリ26は、マスター・プロセッサ12及び共同プロセッサ14がそれらの間でデータを転送すること、ポート18を介して遠隔装置からデータを転送すること、ポート20を介して遠隔装置にデータを転送することを可能としている。またマスター・プロセッサ12及び共同プロセッサ14は、マシン10が生データを処理する速度を制御する共通クロック信号を受け取る。

【0005】

一般に、計算マシン10は、マスター・プロセッサ12及び共同プロセッサ14の間で生データの処理を効果的に分割する。ソナー・アレイ等の遠隔ソース（図1では不図示）は、ポート18を介して、生データに対する先入れ先出し（FIFO）バッファ（不図示）として作用する共有メモリ26の1つの区分に生データをロードする。マスター・プロセッサ12はバス16を介してメモリ26から生データを検索して、マスター・プロセッサ及び共同プロセッサ14はその生データを処理して、バス16を介して必要に応じてデータをそれらの間に転送する。マスター・プロセッサ12はその処理データを共有メモリ26内に規定された別のFIFOバッファ（不図示）にロードし、遠隔ソースがポート20を介してこのFIFOからその処理データを検索する。

【0006】

演算例において、計算マシン10は生データに対する $n+1$ 個の各演算を順次実行することによって該生データを処理し、これら演算は一体的に高速フーリエ変換（FFT）等の処理アルゴリズムを構成する。より詳細には、マシン10はマスター・プロセッサ12及び共同プロセッサ14からのデータ処理パイプラインを形成する。クロック信号の所与の周波数で、そうしたパイプラインはしばしばマシン10が単一プロセッサのみを有するマシンよりも高速に生データを処理することを可能としている。

【0007】

メモリ26内における生データFIFO（不図示）からの生データ検索後、マスター・プロセッサ12はその生データに対して三角関数等の第1番演算を実行する。この演算は第1番結果を生み出し、それをプロセッサ12がメモリ26内に規定された第1番結果FIFO（不図示）に記憶する。典型的には、プロセッサ12はメモリ22内に記憶された

プログラムを実行し、そのプログラムの制御の下で上述した動作を実行する。プロセッサ 12 はメモリ 22 を作業メモリとしても使用し得て、当該プロセッサが第 1 番演算の中間期間に生成するデータを一時的に記憶する。

【0008】

次に、メモリ 26 内における第 1 番結果 FIFO (不図示) からの第 1 番結果検索後、共同プロセッサ 14₁ はその第 1 番結果に対して対数関数等の第 2 番演算を実行する。この第 2 番演算は第 2 番結果を生み出し、それを共同プロセッサ 14₁ がメモリ 26 内に規定された第 2 番結果 FIFO (不図示) に記憶する。典型的には、共同プロセッサ 14₁ はメモリ 24₁ 内に記憶されたプログラムを実行し、そのプログラムの制御の下で上述した動作を実行する。共同プロセッサ 14₁ はメモリ 24₁ を作業メモリとしても使用し得て、当該共同プロセッサが第 2 番演算の中間期間に生成するデータを一時的に記憶する。

【0009】

次に共同プロセッサ 24₂、…、24_n は、共同プロセッサ 24₁ に対して先に議論されたものと同様に、(第 2 番結果—第 (n-1) 番) 結果に対して (第 3 番演算—第 n 番) 演算を順次実行する。

【0010】

共同プロセッサ 24_n によって実行される第 n 番演算は最終結果、即ち処理データを生み出す。共同プロセッサ 24_n はその処理データをメモリ 26 内に規定された処理データ FIFO (不図示) 内にロードし、遠隔装置 (図 1 では不図示) がこの FIFO からその処理データを検索する。

【0011】

マスター・プロセッサ 12 及び共同プロセッサ 14 は処理アルゴリズムの種々の演算を同時に実行するので、計算マシン 10 は、しばしば、種々の演算を順次実行する単一プロセッサを有する計算マシンよりも生データを高速に処理することができる。詳細には、単一プロセッサは、生データから成る先行集合に対する全 (n+1) 個の演算を実行するまで、生データから成る新しい集合を検索できない。しかし、以上に議論したパイプライン技術を用いて、マスター・プロセッサ 12 は第 1 演算だけを実行後に生データから成る新しい集合を検索できる。結果として、所与のクロック周波数でこのパイプライン技術は、単一プロセッサ・マシン (図 1 では不図示) と比較して約 n+1 倍だけマシン 10 が生データを処理する速度を増大することができる。

【0012】

代替的には、計算マシン 10 は、生データに対する FFT 等の処理アルゴリズムの (n+1) 個を同時に実行することによって該生データを並列して処理し得る。即ち、もしそのアルゴリズムが先行する例において先に記載されたような (n+1) 個の順次演算を含めば、マスター・プロセッサ 12 及び共同プロセッサ 14 の各々は生データからそれぞれが成る各集合に対して、順次、全 (n+1) 個の演算を実行する。その結果として、所与のクロック周波数で、先のパイプライン技術と同様のこの並列処理技術は、単一プロセッサ・マシン (図 1 では不図示) と比較して約 n+1 倍だけマシン 10 が生データを処理する速度を増大することができる。

【0013】

残念ながら、計算マシン 10 は単一プロセッサ・計算マシン (図 1 では不図示) と比べてより迅速にデータを処理できるが、マシン 10 のデータ処理速度はしばしばプロセッサ・クロックの周波数より非常に小さい。詳細には、計算マシン 10 のデータ処理速度はマスター・プロセッサ 12 及び共同プロセッサ 14 がデータ処理するのに必要な時間によって制限される。簡略化のため、この速度制限の例はマスター・プロセッサ 12 と連関して議論されているが、この議論は共同プロセッサ 14 にも適用されることを理解して頂きたい。先に議論されたように、マスター・プロセッサ 12 は所望の方式でデータを操作すべくプロセッサを制御するプログラムを実行する。このプログラムはプロセッサ 12 が実行する複数の命令から成るシーケンスを含む。残念ながら、プロセッサ 12 は典型的には単一命令を実行するために多数のクロック・サイクルを必要とし、そしてしばしばデータの

単一値を処理すべく多数の命令を実行しなければならない。例えば、プロセッサ12が第1データ値A（不図示）を第2データ値B（不図示）で乗算することを仮定する。第1クロック・サイクル中、プロセッサ12はメモリ22から乗算命令を検索する。第2及び第3クロック・サイクル中、プロセッサ12はメモリ26からA及びBをそれぞれ検索する。第4クロック・サイクル中、プロセッサ12はA及びBを乗算し、そして第5クロック・サイクル中に結果としての積をメモリ22或は26に記憶するか、或は、その結果としての積を遠隔装置（不図示）に提供する。これは最良ケースのシナリオであり、その理由は多くの場合にプロセッサ12はカウンタの初期化及び閉鎖等のオーバーヘッド・タスクに対して付加的なクロック・サイクルを必要とするからである。それ故に、よくてもプロセッサ12はA及びBを処理すべく5クロック・サイクルを必要とするか、或は、1データ値当たり平均2.5クロック・サイクルを必要とする。

10

【0014】

結果として、計算マシン10がデータを処理する速度は、しばしば、マスター・プロセッサ12及び共同プロセッサ14を駆動するクロックの周波数より非常に低い。例えば、もしプロセッサ12は1.0ギガヘルツ（GHz）でクロックされるが、1データ値当たり平均2.5クロック・サイクルを必要とすれば、効果的なデータ処理速度は（1.0GHz）／2.5＝0.4GHzと同等である。この効果的なデータ処理速度は、しばしば、1秒当たり演算数の単位で特徴付けされる。それ故に、この例において、1.0GHzのクロック速度で、プロセッサ12は0.4ギガ演算数／秒（Gops）で使用限界が定められる。

20

【0015】

図2は、所与クロック周波数で且つしばしば該パイプラインがクロックされる速度と略同一速度で、プロセッサが可能であるよりは高速で典型的にはデータを処理できるハードウェアに組み込まれたデータ・パイプライン30のブロック線図である。パイプライン30は、プログラム命令を実行することなく、各データに対する各演算を各々が実行する演算子回路32₁、…、32_nを含む。即ち、所望の演算は回路32内に「書き込み」が為されて、それがプログラム命令の必要性なしに自動的にその演算を具現化するように為す。プログラム命令の実行と関連されたオーバーヘッドを減ずることによって、パイプライン30は所与のクロック周波数でプロセッサが可能であるよりは単位秒当たりより多くの演算を典型的には実行する。

30

【0016】

例えば、パイプライン30は所与のクロック周波数でプロセッサが可能であるよりは高速で以下の数式1をしばしば解くことができる。

$$Y(x_k) = (5x_k + 3)2^{nk}$$

ここで、 x_k は複数の生データ値から成るシーケンスを表す。この例において、演算子回路32₁は $5x_k$ を計算する乗算器であり、回路32₂は $5x_k + 3$ を計算する加算器であり、そして回路32_n（ $n=3$ ）は $(5x_k + 3)2^{nk}$ を計算する乗算器である。

【0017】

第1クロック・サイクル $k=1$ 中、回路32₁はデータ値 x_1 を受け取って、それを5で乗じて、 $5x_1$ を生成する。

40

【0018】

第2クロック・サイクル $k=2$ 中、回路32₂は回路32₁から $5x_1$ を受け取って、3を加えて、 $5x_1 + 3$ を生成する。またこの第2クロック・サイクル中に回路32₁は $5x_2$ を生成する。

【0019】

第3クロック・サイクル $k=3$ 中、回路32₃は回路32₂から $5x_1 + 3$ を受け取って、 2^{n1} で乗じて（効果としては、 x_1 だけ $5x_1 + 3$ を左シフトする）、第1結果 $(5x_1 + 3)2^{n1}$ を生成する。またこの第3クロック・サイクル中に回路32₁は $5x_3$ を生成し、回路32₂は $5x_2 + 3$ を生成する。

【0020】

50

このようにしてパイプライン30は、全ての生データ値が処理されるまで、引き続き生データ値 x_i の処理を続行する。

【0021】

結果として、生データ値 x_i の受け取り後の2つのクロック・サイクルの遅延、即ち、この遅延はパイプライン30の待ち時間としばしば呼称され、パイプラインは結果 $(5x_1+3)2^{x_1}$ を生成し、その後、1つの結果を生成する、即ち各クロック・サイクル毎に $(5x_2+3)2^{x_2}$ 、 $(5x_3+3)2^{x_3}$ 、 \dots 、 $(5x_n+3)2^{x_n}$ を生成する。

【0022】

待ち時間を無視して、パイプライン30はこうしてクロック速度と同等のデータ処理速度を有する。比較して、マスター・プロセッサ12及び共同プロセッサ14（図1）が先の例におけるようにクロック速度の0.4倍であるデータ処理速度を有すると仮定すれば、パイプライン30は、所与のクロック速度で、計算マシン10（図1）よりも2.5倍高速でデータを処理できる。

【0023】

更に図2で参照されるように、設計者はフィールド・プログラマブル・ゲート・アレイ（FPGA）等のプログラマブル・ロジックIC（PLIC）にパイプライン30を具現化することを選ぶ可能性があり、その理由はPLICが特殊用途IC（ASIC）が為すよりも多くの設計及び変更の柔軟性を許容するからである。PLIC内にハードウェアに組み込まれた接続を構成するため、設計者はPLIC内に配置された相互接続構成レジスタを単に所定バイナリ状態に設定する。全てのこうしたバイナリ状態の組み合わせはしばしば「ファームウェア」と呼称される。典型的には、設計者はこのファームウェアをPLICと結合された不揮発性メモリ（図2では不図示）内にロードする。PLICを「ターンオン」すると、それはファームウェアをそのメモリから相互接続構成レジスタにダウンロードする。それ故に、PLICの機能を変更すべく、設計者は単にそのファームウェアを変更して、PLICがその変更されたファームウェアを相互接続構成レジスタにダウンロードすることを可能とする。ファームウェアを単に変更することによってPLICを変更する能力は、モデル作成段階中や「フィールド内」にパイプライン30をアップグレードするために特に有用である。

【0024】

残念ながら、ハードウェアに組み込まれたパイプライン30は重要な意思決定、特に入れ子意思決定を引き起こすアルゴリズムを実行すべき最良の選択でない可能性がある。プロセッサは、典型的には、入れ子意思決定命令（例えば、「もしAであれば、Bを為し、またもしCであれば、Dを為し、 \dots 、またnを為し等々」のように、入れ子条件命令）を、比肩する長さの演算命令（例えば、「 $A+B$ 」）を実行できる程に高速に実行できる。しかしパイプライン30は、比較的単純な決定（例えば、「 $A>B?$ 」）を効率的に為し得るが、典型的にはプロセッサができる程に効率的に入れ子決定（例えば、「もしAであれば、Bを為し、またもしCであれば、Dを為し、 \dots 、またnを為す」）を為すことができない。この非効率性の1つの理由は、パイプライン30はほんの僅かなオンボード・メモリしか持たないことがあり、したがって外部作業／プログラム・メモリ（不図示）にアクセスすることを必要とすることがあるからである。そして、こうした入れ子決定を実行すべくパイプライン30を設計することができるが、必要とされる回路のサイズ及び複雑性はしばしばそうした設計を非現実的に為し、特にアルゴリズムが多数の種々の入れ子決定を含む場合にそうである。

【0025】

結果として、プロセッサは典型的には重要な意思決定を必要とする用途において使用され、ハードウェアに組み込まれたパイプラインは殆ど意思決定が為されないか或は意思決定されない「ナンバークランピング（数値データ処理）」用途に典型的には限定される。

【0026】

更には、下記に議論されるように、典型的には、特にパイプライン30が多数のPLICを含む場合、図2のパイプライン30等のハードウェアに組み込まれたパイプラインを

設計／変更するよりも、図1の計算マシン10等のプロセッサに基づく計算マシンを設計／変更することが非常に易しい。

【0027】

プロセッサ及びそれらの周辺機器（例えば、メモリ）等の計算構成要素は、典型的には、プロセッサに基づく計算マシンを形成すべくそれら構成要素の相互接続を補助する工業規格通信インターフェースを含む。

【0028】

典型的には、規格通信インターフェースは2つの層、即ち、物理層及びサービス層を含む。

【0029】

物理層は、回路とこの回路のインターフェース及び動作パラメータを形成する対応回路相互接続とを含む。例えば、物理層はそれら構成要素を1つのバスに接続するピンと、それらのピンから受け取ったデータをラッチするバッファと、信号をそれらピンに駆動するドライバと、入力データ信号からデータを回復すると共にそのデータ信号或は外部クロック信号からクロック信号を回復する回路とを含む。動作パラメータは、ピンが受け取るデータ信号の許容可能電圧範囲と、データの書き込み及び読み取りのための信号タイミングと、動作の支援されたモード（例えば、バーストモード、ページモード）とを含む。従来の物理層はトランジスタートランジスタ論理（TTL）及びRAMBUSを含む。

【0030】

サービス層は、計算構成要素のデータ転送のためのプロトコルを含む。このプロトコルはデータのフォーマットと、構成要素によるフォーマット済みデータの送受信の方式とを含む。従来の通信プロトコルは、ファイル転送プロトコル（FTP）及び伝送制御プロトコル／インターネット・プロトコル（TCP/IP）を含む。

【0031】

結果として、製造業者やその他は工業規格通信インターフェースを有する計算構成要素を典型的には設定するので、そうした構成要素のインターフェースを典型的には設計でき、それを他の計算構成要素と比較的少ない労力で相互接続することができる。これは、計算マシンの他の部分の設計に設計者自信の時間を殆ど費やすことを可能として、各種構成要素を追加或は除去することによってそのマシンを変更することを可能としている。

【0032】

工業規格通信インターフェースを支援する計算構成要素を設計することは、設計ライブラリから既存の物理層を用いることによって設計時間を節約することを可能としている。これは、設計者が構成要素を既製の計算構成要素と容易にインターフェースすることを保証するものである。

【0033】

そして、共通した工業規格通信インターフェースを支援する計算構成要素を用いる計算マシンを設計することは、設計者がそれら構成要素を少しの時間及び労力で相互接続することを可能としている。それら構成要素は共通インターフェースを支援するので、設計者はそれらをシステム・バスを介して少しの設計労力で相互接続することができる。そして、その支援されたインターフェースは工業規格であるので、マシンを容易に変更することができる。例えば、システム設計が進化するに伴って種々の構成要素及び周辺機器をマシンに追加することができるか、或は、テクノロジーが進化するに伴って次世代の構成要素を追加／設計することが可能である。更には、構成要素が通常の工業規格サービス層を支援するので、計算マシンのソフトウェアに対応するプロトコルを具現化する既存のソフトウェア・モジュールを組み込むことができる。それ故に、インターフェース設計が本質的には既に整っているのので少しの労力で構成要素をインターフェースでき、よって、マシンに所望の機能を実行させるマシンの各種部分（例えばソフトウェア）の設計に集中することができる。

【0034】

しかし残念ながら、図2のパイプライン30等のハードウェアに組み込まれたパイプラ

インを形成すべく、使用される P L I C 等の各種構成要素に対する既知の工業規格サービス層が全くない。

【0035】

結果として、多数の P L I C を有するパイプラインを設計すべく、多大な時間を費やし、「ゼロから」設計し且つ種々の P L I C の間の通信インターフェースのサービス層をデバッグする多大な労力を行行使する。典型的には、そうしたその場限りのサービス層は種々の P L I C 間で転送されるデータのパラメータに依存する。同じように、プロセッサとインターフェースするパイプラインを設計すべく、パイプライン及びプロセッサの間の通信インターフェースのサービス層の設計及びデバッグに関して多大な時間を費やし且つ多大な労力を行行使する必要がある。

10

【0036】

同様に、そうしたパイプラインを P L I C を該パイプラインに追加することによって変更すべく、典型的には、その追加された P L I C と既存の P L I C との間の通信インターフェースのサービス層の設計及びデバッグに関して多大な時間を費やし且つ多大な労力を行行使する。同じように、プロセッサを追加することによってパイプラインを変更すべく、或は、パイプラインを追加することによって計算マシンを変更すべく、パイプライン及びプロセッサの間の通信インターフェースのサービス層の設計及びデバッグに関して多大な時間を費やし且つ多大な労力を行行使しなければならないであろう。

【0037】

結果として、図 1 及び図 2 で参照されるように、多数の P L I C をインターフェースすることとプロセッサをパイプラインにインターフェースすることとの難しさのため、計算マシンを設計する際に多大な妥協を為すことがしばしば強いられる。例えば、プロセッサに基づく計算マシンでは、ナンバークランピング速度を、複雑な意思決定を為す能力に対する設計/変更の柔軟性と交換することを強いられる。逆に、ハードウェアに組み込まれたパイプラインに基づく計算マシンでは、複雑な意思決定を為す能力と設計/変更の柔軟性を、ナンバークランピング速度と交換することを強いられる。更には、多数の P L I C をインターフェースすることに関する難しさのため、少数の P L I C よりも多くの P L I C を有するパイプラインに基づくマシンを設計することはしばしば実際的ではない。その結果、実際的なパイプラインに基づくマシンはしばしば制限された機能しか有さない。そして、プロセッサを P L I C とインターフェースすることに関する難しさのため、プロセッサを 1 つの P L I C より多くの P L I C にインターフェースすることは実際的ではない。その結果、プロセッサ及びパイプラインを組み合わせることによって獲得される利益は最少となる。

20

30

【発明の開示】

【発明が解決しようとする課題】

【0038】

それ故に、プロセッサに基づくマシンの意思決定を為す能力を、ハードウェアに組み込まれたパイプラインに基づくマシンのナンバークランピング速度と組み合わせることを可能とする新しい計算アーキテクチャに対する要望が生じてきている。

【課題を解決するための手段】

40

【0039】

本発明の実施例に従えば、プログラマブル回路は外部ソースからファームウェアを受信し、そのファームウェアをメモリ内に記憶してから、そのファームウェアをそのメモリからダウンロードする。

【0040】

そうしたプログラマブル回路は、計算マシン等のシステムにプログラマブル回路のコンフィギュレーションを変更させることを可能とし、よってコンフィギュレーション・メモリをマニュアルで再プログラムする必要性を削減している。例えば、もしプログラマブル回路がパイプライン加速器の部分である F P G A であれば、その加速器と結合されたプロセッサは F P G A のコンフィギュレーションを変更できる。より詳細には、プロセッサは

50

コンフィギュレーション・レジストリからその変更されたコンフィギュレーションを表すファームウェアを検索して、そのファームウェアをFPGAに送信し、次いで該FPGAがそのファームウェアを、電氣的に消去可能でプログラム可能な読み取り専用（EEPROM）等のメモリ内に記憶する。次に、FPGAはそのメモリからそのコンフィギュレーション・レジストリにファームウェアをダウンロードし、こうして変更されたコンフィギュレーションを有するようにそれ自体を効果的に再構成する。

【発明を実施するための最良の形態】

【0041】

図3は、本発明の一実施例に従ったピア-ベクトル・アーキテクチャを有する計算マシン40の概略ブロック線図である。ホストプロセッサ42に加えて、ピア-ベクトル・マシン40はパイプライン加速器44を含み、それがデータ処理の少なくとも一部を実行して、図1の計算マシン10における共同プロセッサ14の列と効果的に置き換わる。それ故に、ホストプロセッサ42及び加速器44（又は以下に議論されるようにそのパイプライン・ユニット）はデータ・ベクトルを前後に転送できる「ピア」である。加速器44がプログラム命令を実行しないので、所与のクロック周波数で共同プロセッサの列ができるものよりも著しく高速にデータに対して数学的に集中的な演算を典型的には実行する。結果として、プロセッサ42の意思決定能力と加速器44のナンバークランピング能力とを組み合わせることによって、マシン40はマシン10等の従来の計算マシンと同一の能力を有するが、しばしばそれよりもデータをより高速に処理することができる。更には、以下に議論されるように、加速器44にホストプロセッサ42の通信インターフェースと互換性がある通信インターフェースを設けることが、特にプロセッサの通信インターフェースが工業規格である場合に、マシン40の設計及び変更を補助する。そして、加速器44が1つ或はそれ以上のPLICを含む場合、ホストプロセッサ42は適切なファームウェアをそれらPLICに送信することによって加速器内における物理的な相互接続コネクタをハード的に構成できる。ホストプロセッサ42はピア-ベクトル・マシン40の初期化中にこの方式で加速器44を構成できるばかりではなく、以下に議論されると共に先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献3に議論されるように、ピア-ベクトル・マシンの動作中に加速器を再構成する能力を有し得る。更には、ピア-ベクトル・マシン40は以下に議論されると共に先行して引用された特許出願におけるような他の長所等をも提供し得る。

【0042】

更に図3で参照されるように、ホストプロセッサ42及びパイプライン加速器44に加えて、ピア-ベクトル・計算マシン40は、プロセッサ・メモリ46、インターフェース・メモリ48、パイプライン・バス50、1つ或はそれ以上のファームウェア・メモリ52、任意選択的な生データ入力ポート54、処理済みデータ出力ポート58、任意選択的なルータ61、並びに、試験バス63を含む。

【0043】

ホストプロセッサ42は処理ユニット62及びメッセージ・ハンドラー64を含み、プロセッサ・メモリ46は処理ユニット・メモリ66及びハンドラー・メモリ68を含み、そのそれぞれがプロセッサ・ユニット及びメッセージ・ハンドラーに対するプログラム及び作業の両メモリとして役立っている。プロセッサ・メモリ46は、加速器コンフィギュレーション・レジストリ70及びメッセージ・コンフィギュレーション・レジストリ72をも含み、それらが、ホストプロセッサ42に加速器44の機能とメッセージ・ハンドラー64が送信及び受信するメッセージのフォーマットと構成させることをそれぞれ可能とするファームウェア及びコンフィギュレーション・データを記憶する。加速器44及びメッセージ・ハンドラー64のコンフィギュレーションは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献3に更に議論され、加速器44のコンフィギュレーションも図4乃至図6と連携して以下に更に議論される。

【0044】

パイプライン加速器 44 は少なくとも 1 つの P L I C (図 4) 上に配置され、プログラム命令を実行することなしに各データを処理するハードウェアに組み込まれたパイプライン 74、ー 74、を含む。ファームウェア・メモリ 52 は加速器 44 に対するファームウェアを記憶する。より詳細には、ファームウェア・メモリ 52 は、図 4 乃至図 6 と連携して以下に更に議論されるように、加速器 44 を構成する複数の P L I C に対するファームウェアを記憶する。代替的には、加速器 44 は少なくとも 1 つの A S I C 上に配置され得て、その A S I C がひとたび形成されたならば構成不可能である内部相互接続を有し得る。加速器 44 が P L I C を何等含まないこの代替例において、マシン 40 はファームウェア・メモリ 52 を省略し得る。更には、加速器 44 が多数パイプライン 74、ー 74、を含んで示されているが、ただ 1 つのパイプラインを含み得る。加えて、図示されていないが、加速器 44 はデジタル信号プロセッサ (D S P) 等の 1 つ或はそれ以上のプロセッサを含み得る。更には、図示されていないが、加速器 44 はデータ入力ポート及び/或はデータ出力ポートを含み得る。

【 0045 】

ビームベクトル・マシン 40 の一般動作は、先行して引用された「改善された計算アーキテクチャ、関連システム、並びに、方法」と題された特許文献 2 に議論されており、ホストプロセッサ 42 の構造及び動作は、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献 3 に議論されており、パイプライン加速器 44 の構造及び動作は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献 4 及び「多数パイプライン・ユニットを有するパイプライン加速器、関連計算マシン、並びに、方法」と題された特許文献 5 に議論されている。加速器 44 を構成する P L I C の動作コンフィギュレーションは、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献 4 や、図 4 乃至図 6 と連携して以下に議論されている。

【 0046 】

図 4 乃至図 6 で参照されるように、加速器 44 P L I C を「ハード」的に構成するための技術が議論される。上記に既に触れているように、P L I C のハード・コンフィギュレーションはファームウェアによってプログラムされ、P L I C の種々の構成要素間での特定の物理的相互接続、即ち、どのようにして一方の論理ブロックが他方の論理ブロックと電気的に接続されているかを示す。これは、既にハード的に構成された P L I C のより高レベル・コンフィギュレーションを示す「ソフト」コンフィギュレーションとは対照的である。例えば、ハード的に構成された P L I C はバッファを含み得ると共にレジスタをも含み得て、そのレジスタによって対応するソフト・コンフィギュレーション・データを該レジスタにロードすることによって、そのバッファのサイズをソフト的に構成させることができる。加速器 44 のソフト・コンフィギュレーションは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献 3 及び「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献 4 に更に議論されている。

【 0047 】

図 4 は、本発明の実施例に従った図 3 のパイプライン加速器 44 のパイプライン・ユニット 78 のブロック線図である。ハードウェアに組み込まれたパイプライン 74、ー 74、(図 3) はパイプライン・ユニット 78 の部分であり、それは、以下に議論されるように、例えば、ハードウェアに組み込まれたパイプラインを制御すると共にそれらにデータを受信、送信、そして記憶させることを可能とする回路を含む。1 つのみのパイプライン・ユニット 78 が図 4 に示されているが、加速器 44 は、先行して引用された「多数パイプライン・ユニットを有するパイプライン加速器、関連計算マシン、並びに、方法」と題された特許文献 5 に議論されたように、多数のパイプライン・ユニット (各々がハードウェアに組み込まれたパイプライン 74、ー 74、の内の少なくとも幾つかを含む) を含み得る。以下に議論されるように、1 つの具現化例において、パイプライン・ユニット 78 のハ

ード・コンフィギュレーションはファームウェアでプログラム可能である。これは、ファームウェアを単に変更することによって、パイプライン・ユニット78の機能を変更させることを可能としている。更には、ホストプロセッサ42（図3）はビームベクトル・マシン40（図3）の初期化中或は再構成中に変更されたファームウェアをパイプライン・ユニット78に提供でき、よってその変更ファームウェアをパイプライン・ユニットにマニユアルでロードさせる必要性をなくせる。

【0048】

パイプライン・ユニット78は、PLIC或はASIC等のパイプライン回路80、ファームウェア・メモリ52（パイプライン回路がPLICの場合）、並びに、データ・メモリ81を含み、それら全てが回路ボード或はカード83上に配置されることになる。データ・メモリ81は、先行して引用された「プログラマブル回路、関連計算マシン、並びに、方法」と題された米国特許出願第10/684,057号に更に議論され、パイプライン回路80及びファームウェア・メモリ52の組み合わせはプログラマブル回路ユニットを形成する。

【0049】

パイプライン回路80は通信インターフェース82を含み、それが、ホストプロセッサ42（図3）等のピアとデータ・メモリ81との間、そして、ピアと、通信シェル84を介したハードウェアに組み込まれたパイプライン74、74_a、パイプライン・コントローラ86、例外マネージャ88、並びに、コンフィギュレーション・マネージャ90等の、パイプライン回路の他の構成要素との間でデータを転送する。パイプライン回路80は工業規格バス・インターフェース91及びインターフェース82をインターフェース91と接続する通信バス93をも含み得る。代替的には、インターフェース91の機能は通信インターフェース82内に含まれ得る。バス93は省略され得る。ハードウェアに組み込まれたパイプライン74、74_a、コントローラ86、例外マネージャ88、コンフィギュレーション・マネージャ90、並びに、バス・インターフェース91の構造及び動作は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献4に議論されている。

【0050】

通信インターフェース82はメッセージ・ハンドラー64（図3）によって認識されるフォーマットでデータを（存在する場合にはバス・インターフェース91を介して）送受信し、よってピアベクトル・マシン40（図3）の設計及び変更を典型的には補助する。例えば、もしデータ・フォーマットが高速1/0フォーマット等の工業規格であれば、ホストプロセッサ42及びパイプライン・ユニット78の間にカスタムインターフェースを設計する必要がない。更には、パイプライン・ユニット78に非バス・インターフェースの代わりにパイプライン・バス50を介してホストプロセッサ42（図3）等の他のピアと通信させることを可能することによって、パイプライン・ユニットが追加或は除去されるたびにスクラッチから非バス・インターフェースを再設計する代わりに、パイプライン・バスにそれらパイプライン・ユニット（又はそれらを保持する回路カード）を単に接続或は接続解除することによってパイプライン・ユニットの数を変更できる。

【0051】

パイプライン回路80がFPGA等のPLICである場合、通信インターフェース82はプログラマブル・ポート94を含み、それが以下に議論されるようにパイプライン回路にホストプロセッサ42（図3）からファームウェア・メモリ52にファームウェアをロードさせることを可能としている。例えば、もしファームウェア・メモリ52がEEPROMであるならば、プログラム・サイクル中にファームウェア・メモリが必要とするプログラミング信号を、通信インターフェース82は生成し、ポート94は配送する。そうしたプログラミング信号を生成する回路は従来のものであり、よって更に議論されない。

【0052】

通信インターフェース82の構造及び動作は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献

4に更に議論されている。

【0053】

更に図4で参照されるように、パイプライン回路80は、試験ポート96、パイプライン回路がPLICである場合にハード・コンフィギュレーション・ポート98をも含む。試験バス63と結合された試験ポート96は、ホストプロセッサ42（図3）に、以下に議論されるように、パイプライン回路80がビタベクトル・マシン40（図3）の初期化中に実行し得る自己試験の結果をモニタさせることを可能とする。製品には、典型的には、パイプライン回路80を伴う試験ポート96を含み、そして典型的には、試験ポートにJTAG等の工業規格試験プロトコルと互換性があるインターフェース（不図示）を提供する。ハード・コンフィギュレーション・ポート98は、以下に議論されるように、パイプライン回路80にメモリ52からファームウェアをダウンロードすることによってそれ自体を構成させることを可能とする。試験ポート96のように、製品には、典型的には、パイプライン回路80を伴うコンフィギュレーション・ポート98を含み、そして典型的にはコンフィギュレーション・ポートに工業規格メモリ・インターフェースと、メモリ52の所定アドレス範囲からファームウェアを順次ダウンロードする状態マシンと（双方とも不図示）を提供する。

【0054】

先に議論されると共に以下に更に議論されるように、パイプライン回路80がPLICである場合、ファームウェア・メモリ52はパイプライン回路の1組或はそれ以上の組みのハード・コンフィギュレーションを表すファームウェアを記憶する。このファームウェア・メモリ52は試験ポート104と、プログラミング及びコンフィギュレーション・ポート106、108とを含む。試験バス63と結合された試験ポート104は、ホストプロセッサ42（図3）に、以下に議論されるように、ファームウェア・メモリ52がビタベクトル・マシン40（図3）の初期化中に実行し得る自己試験の結果をモニタさせることを可能とする。また以下に議論されるように、試験ポート104はホストプロセッサ42にファームウェアをメモリ52にロードさせることを可能とし得る。製品には、典型的には、メモリ52を伴う試験ポート104を含み、そして典型的にはその試験ポートにJTAG等の工業規格試験プロトコルと互換性があるインターフェース（不図示）を提供する。プログラミング・バス110を介して通信インターフェース82のプログラミング・ポート94と結合されたプログラミング・ポート106は、以下に議論されるようにパイプライン回路80にファームウェアをメモリ52にロードさせることを可能とする。そして、コンフィギュレーション・バス112を介してパイプライン回路80のハード・コンフィギュレーション・ポート98と結合されたハード・コンフィギュレーション・ポート108は、以下に議論されるように、パイプライン回路にメモリ52からファームウェアをダウンロードさせることを可能とする。典型的には、ファームウェア・メモリ52はEEPROM等の不揮発性メモリであり、電力が欠如している状態でデータを保持する。結果として、ファームウェア・メモリ52はパイプライン・ユニット78が電力がダウンした後もそのファームウェアを記憶し続ける。

【0055】

更に図4に参照されるように、ファームウェア・メモリ52及びデータ・メモリ81がパイプライン回路80の外部であると説明されているが、何れか一方或は両方のメモリはパイプライン回路に組み込まれ得る。メモリ52がパイプライン回路80の内部に配置されている場合、設計者はそれに対応してプログラミング及びコンフィギュレーション・バス110、112の構造を変更する必要がある。更には、パイプライン・ユニット78がコンフィギュレーション・バス112から分離したプログラミング・バス110を有すると説明されているが、単一バス（不図示）はプログラミング及びコンフィギュレーションの両バスの機能を実行し得る。代替的には、パイプライン・ユニット78はこの単一バスの多数例を含み得るか、或は、プログラミング及びコンフィギュレーション112、110の一方或は両方の多数例を含み得る。

【0056】

図5は、本発明の実施例に従った図4のファームウェア・メモリ52の論理区画の線図である。

【0057】

メモリ52の区分114はパイプライン回路80（図4）の初期コンフィギュレーションを表すファームウェアを記憶する。即ち、パイプライン回路80にダウンロードされると、このファームウェアはパイプライン回路に初期コンフィギュレーションを所有させる。初期コンフィギュレーションの1つの具現化例において、パイプライン回路80は図4の通信インターフェース82（そしてもし必要があれば工業規格バス・インターフェース91）と、パイプライン回路及びデータ・メモリ81の自己試験を実行する自己試験回路（不図示）を含む。パイプライン回路80は、次いで、試験バス63或は通信インターフェース82を介して自己試験の結果をホストプロセッサ42（図3）に提供できる。また初期コンフィギュレーションは、ホストプロセッサ42に変更されたファームウェアを、以下に議論されるように、通信インターフェース82及びプログラミング・バス110を介してファームウェア・メモリ52にロードさせることを可能としている。

【0058】

メモリ52の区分116₁〜116_nは、各々、パイプライン回路80の各動作コンフィギュレーションを表すファームウェアを記憶する。典型的には、パイプライン回路80は、加速器44（図3）の初期化の最後に区分116₁〜116_nの内の所定の1つからファームウェアをダウンロードする。以下に議論されるように、パイプライン回路80は特定の区分116₁〜116_nからファームウェアをダウンロードすべく予めプログラムされ得るか、或は、ホストプロセッサ42（図3）はパイプライン回路に特定の区分からファームウェアをダウンロードするように命令する。典型的には、1個の動作コンフィギュレーションの各々において、パイプライン回路80は図4に示される構成要素（例えば、ハードウェアに組み込まれたパイプライン74₁〜74_n、コントローラ86）を含む。しかし、これらコンフィギュレーションの各々において、パイプライン回路80は典型的には異なるように動作する。例えば、通信インターフェース82は一方のコンフィギュレーションにおいて一方のプロトコルを具現化し、他方のコンフィギュレーションにおいて他方のプロトコルを具現化する。或は、パイプライン74₁〜74_nは一方のコンフィギュレーションにおいてデータに対して一方の組の演算を実行し得て、他方のコンフィギュレーションにおいてデータに対して他方の組の演算を実行し得る。

【0059】

任意選択的な区分118は、メモリ52の区分116₁〜116_nに記憶されたファームウェアによってそれぞれ表される動作コンフィギュレーションの記述或は識別表示を記憶する。この記述／識別表示はホストプロセッサ42（図3）にメモリ52に記憶されたファームウェアを識別させることを可能とする。

【0060】

任意選択的な区分120はパイプライン・ユニット78（図4）のプロファイルを記憶する。このプロファイルはパイプライン・ユニット78のハードウェア・レイアウトを十分に記述して、ホストプロセッサ42（図3）が、それ自体、パイプライン・ユニット、並びに、ピア・ベクトル・マシン40（図3）の別のピア（不図示）を相互通信のために適切に構成するように為す。例えばプロファイルは、パイプライン・ユニット78が具現化できるデータ演算及び通信プロトコル、データメモリ81のサイズ、区分116₁〜116_nに記憶されたファームウェアによって表される動作コンフィギュレーション（もし区分118が省略された場合）、並びに、現行において所望された動作コンフィギュレーションを識別し得る。結果として、ピア・ベクトル・マシン40の初期化中にプロファイルを読み取ることによって、ホストプロセッサ42は、パイプライン・ユニット78と通信すべく、メッセージ・ハンドラー64（図3）を適切に構成できる。更には、ホストプロセッサ42はパイプライン回路80がダウンロードすべきファームウェアの区分116₁〜116_nを選択し得る。或は、もしこのファームウェアの何れも適合しなければ、ホストプロセッサ42は変更されたファームウェアをメモリ52にロードし得る。この技術は

「プラグ・アンド・プレイ」技術と類似し、それによってコンピュータはそれ自体を構成できて、新しくインストールされたディスク・ドライブ等の周辺機器と通信する。

【0061】

代替的には、区分120は、ホストプロセッサ42（図3）に、例えば、加速器コンフィギュレーション・レジストリ70（図3）に記憶されているテーブルからプロファイルを検索させることを可能とする（しばしば「実行インデックス」と呼称される）プロファイル識別子を記憶し得る。典型的には、実行インデックスは数であり、製品の型番号と非常に似ており、ホストプロセッサ42は記憶されたプロファイルと照合することができる。

【0062】

更に別の代替例において、パイプライン・ユニット78（図4）は「ハードウェアに組み込まれた」形態でプロファイル識別子を記憶し得て、区分120のプロファイルを故意ではなく上書きし得る機会をなくする。例えば、パイプライン・ユニット78は、ホストプロセッサ42（図3）が試験バス63或はパイプライン・バス50及びパイプライン回路80（図4）を介して読み取ることができるハードウェアに組み込まれた「レジスタ」に、そのプロファイル識別子を記憶し得る。このレジスタは、電気機械式スイッチ、ジャンパー、或は、半田付け接続部（不図示）等で形成され得る。

【0063】

更に図5で参照されるように、ファームウェア・メモリ52の任意選択的な区分122は、ファームウェア・メモリ52が加速器44の初期化中に実行する自己試験ルーチン等のその他のデータを記憶し得る。

【0064】

図3乃至図5で参照されるように、ピアーベクトル・マシン40の動作（特にホストプロセッサ42、パイプライン回路80、並びに、ファームウェア・メモリ52の動作）は、本発明の実施例に従って以下に議論される。

【0065】

ピアーベクトル・マシン40は先ず電源投入されると、ホストプロセッサ42は、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献3に議論されているようにそれ自体を初期化し、加速器44はそれ自体を部分的に初期化する。より詳細には、この部分的初期化中、パイプライン回路80はメモリ52の区分114から初期コンフィギュレーション・ファームウェアをダウンロードする。先に議論されたように、初期コンフィギュレーションにおいて、パイプライン回路80は少なくとも通信インターフェース82及び試験回路（不図示）を含む。パイプライン回路80がその初期コンフィギュレーションで構成された後、試験回路はパイプライン回路及びデータ・メモリ81の自己試験を実行し、その自己試験の結果を試験ポート96及び試験バス63を介してホストプロセッサ42に提供する。ファームウェア・メモリ52も自己試験を実行し得て、図5と連携して先に議論されたように、その結果を試験ポート104及び試験バス63を介してホストプロセッサ42に提供する。

【0066】

次に、ホストプロセッサ42は加速器44の部分的初期化中に例外が生じたかを決定する。例えば、ホストプロセッサ42は試験バス63からの自己試験結果を分析して、パイプライン回路80、データ・メモリ81、並びに、ファームウェア・メモリ52が適切に機能しているかを決定する。

【0067】

もし例外が生じたならば、ホストプロセッサ42はそれを所定方式で取り扱う。例えば、もしホストプロセッサ42がパイプライン回路80から自己試験結果を受信しなければ、試験バス63を介して、初期コンフィギュレーション・ファームウェアがファームウェア・メモリ52の区分114内に記憶されているかを検査し得る。もしその初期コンフィギュレーション・ファームウェアが記憶されていないならば、ホストプロセッサ42はその初期コンフィギュレーション・ファームウェアをパイプライン・バス50或は試験バス6

3を介して区分114にロードして、パイプライン回路80にこのファームウェアをダウンロードさせてから、自己試験の結果を分析し得る。例外のホストプロセッサの取り扱いは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献3に更に議論されている。

【0068】

もし例外が生じなければ、ホストプロセッサ42はパイプライン・ユニット78からプロファイル識別子を読み取り、引き続き加速器コンフィギュレーション・レジストリ70からパイプライン・ユニットの対応するプロファイルを獲得する。ファームウェア・メモリ52の区分120からの代わりに、レジストリ70からプロファイルを獲得することはしばしばより好ましく、その理由は、もしパイプライン回路80がASICであれば、パイプライン・ユニット78がファームウェア・メモリ等の不揮発性メモリを含まないことがあり得るからである。もしそのプロファイル識別子がパイプライン回路80がASICであることを示せば、ホストプロセッサ42はファームウェアがパイプライン回路にダウンロードされる必要性が何もないことを決定する。代替的には、ホストプロセッサ42（図3）はファームウェア・メモリ52の区分120からプロファイルを獲得し得る。この代替例において、プロファイル識別子を記憶することはパイプライン・ユニット78にとって不必要であるが、該パイプライン・ユニットはそのプロファイルが区分120から放逐ではなく削除される場合にプロファイル識別子を記憶し得る。

【0069】

次に、パイプライン・ユニット78（図4に1つのみ示されている）の全てからプロファイル識別子を読み取った後、ホストプロセッサ42は加速器44におけるパイプライン・ユニット78全てのマップを効果的に生成し、そのマップを例えばハンドラー・メモリ68内に記憶する。

【0070】

次いで、各パイプライン・ユニット78に対して、ホストプロセッサ42はプロファイルからパイプライン回路80の所望の動作コンフィギュレーションの識別子を抽出する。加速器44の初期化中に所望の動作コンフィギュレーションを抽出することは、初期化に先行してそのプロファイルを一に更新することによってパイプライン回路80の動作を変更させることを可能とする。

【0071】

次に、ホストプロセッサ42は所望の動作コンフィギュレーションを表すファームウェアがファームウェア・メモリ52に記憶されているかを決定する。例えば、ホストプロセッサ42は、プログラミング・バス110及び（パイプライン回路80が初期コンフィギュレーションにあるので、通信インターフェースが存在するので）通信インターフェース82を介してメモリ区分118からそのコンフィギュレーション記述を読み取ることができ、所望のファームウェアが区分116、116の何れかに記憶されているかを決定する。代替的には、ホストプロセッサ42は試験バス63及び試験ポート104を介してメモリ52から直にコンフィギュレーション記述を読み取り得る。

【0072】

もし所望の動作コンフィギュレーションを表すファームウェアがファームウェア・メモリ52に記憶されていなければ、ホストプロセッサ42は通信インターフェース82、プログラミング・ポート94、106、並びに、プログラミング・バス110を介して加速器コンフィギュレーション・レジストリ70からファームウェア・メモリの区分116、116の内の1つにこのファームウェアをロードする。もしファームウェアがレジストリ70内になければ、ホストプロセッサ42は外部ライブラリ（不図示）からファームウェアを検索し得るか、或は、例外指標を生成し得て、システム動作（不図示）がそのファームウェアをレジストリ70にロードし得るように為す。

【0073】

次に、ホストプロセッサ42は、パイプライン回路80にポート108、コンフィギュレーション・バス112、並びに、ポート98を介してメモリ52の対応する区分116

「116」から所望のファームウェアをダウンロードさせるように命令する。

【0074】

パイプライン回路80が所望のファームウェアをダウンロードした後、所望の動作コンフィギュレーションとなって、データ処理を始める準備が為される。しかし、パイプライン回路80がその所望動作コンフィギュレーションとなった後でさえ、ホストプロセッサ42は通信インターフェース82或は試験バス63を介して新しいファームウェアをメモリ52の区分116、「116」にロードし得る。例えば、新しいファームウェアをロードすべく、ホストプロセッサ42は先ずパイプライン回路80にメモリ52の区分114からファームウェアを再ロードさせ得て、パイプライン回路が再び初期コンフィギュレーションとなるように為す。次いで、ホストプロセッサ42はパイプライン・バス50及び通信インターフェース82を介して新しいファームウェアを区分116、「116」の内の1つにロードする。次に、ホストプロセッサ42はパイプライン回路80にその新しいファームウェアをダウンロードさせて、パイプライン回路がその新しい動作コンフィギュレーションとなるように為す。初期コンフィギュレーションであるときにのみ、パイプライン回路80に新しいファームウェアをメモリ52にロードさせることは、2つの長所をもたらす。第1として、それはパイプライン回路80が動作コンフィギュレーションにある際に該パイプライン回路がメモリ52に記憶されているファームウェアを故意ではなく変更することを防止する。第2として、それは動作コンフィギュレーションに、さもないればファームウェアをメモリ52に書き込むために必要とされる回路用を使用されることになるパイプライン回路80のリソースを利用させることを可能とする。

【0075】

図6は、本発明の別の実施例に従った図3のパイプライン加速器44のパイプライン・ユニット124のブロック図である。

【0076】

パイプライン・ユニット124は、そのパイプライン・ユニット124が多数のパイプライン回路80（ここでは2つのパイプライン回路80a及び80b）と、各パイプライン回路に対して1つずつのメモリである多数のファームウェア・メモリ（ここでは2つのメモリ52a及び52b）と、を含むことを除いて、図4のパイプライン・ユニット78と類似している。パイプライン回路80a及び80bとファームウェア・メモリ52a及び52bとの組み合わせはプログラマブル回路ユニットを形成する。1つの具体化例において、メモリ52a及び52bの各々は、ファームウェア52bが、パイプライン・ユニット124のプロファイルを記憶し、さもないればメモリ52aの区分120と重複するような区分120を省略し得ることを除いては、図5に示されるように仕切られる。代替的には、パイプライン回路80a及び80bはメモリ52a及び52bと動作的に類似している各区分を含む単一ファームウェア・メモリを共有し得る。パイプライン回路80の数を増大することは、典型的には、ハードウェアに組み込まれたパイプライン74、「74」の数nに関する増大を許容し、よってパイプライン・ユニット78と比較してのパイプライン・ユニット124の機能に関する増大を許容する。更には、パイプライン回路80a及び80bの何れか一方或は双方はASICであり得て、その場合、対応するファームウェア・メモリ（又は対応する複数のファームウェア・メモリ）52は省略し得る。

【0077】

パイプライン・ユニット124の構造及び動作の更なる詳細は、先行して引用された「改善された計算アーキテクチャ用パイプライン加速器、関連システム、並びに、方法」と題された特許文献4に議論されている。

【0078】

パイプライン回路80aは試験ポート96a及びハード・コンフィギュレーション・ポート98aを含み、それらが図4の試験ポート96及びハード・コンフィギュレーション・ポート98とそれぞれ類似している。そして図4のパイプライン回路80と同じように、パイプライン回路80aはプログラミング・ポート94を有する通信インターフェース82を含む。

【0079】

パイプライン回路80bは試験ポート96a及びハード・コンフィギュレーション・ポート98bを含み、それらも図4の試験ポート96及びハード・コンフィギュレーション・ポート98とそれぞれ類似している。そして、ホストプロセッサ42（図3）が以下に議論されるようにパイプライン回路80aの通信インターフェース82を介してファームウェア・メモリ52bをプログラムできるので、パイプライン回路80bはプログラミング・ポートを含まない。

【0080】

ファームウェア・メモリ52aは試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート104a、106a、108aを含み、それらが図4の試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート104、106、108とそれぞれ類似している。試験ポート104aは試験バス63と結合され、プログラミング・ポート106aはプログラミング・バス110を介して通信インターフェース82のプログラミング・ポート94aと結合され、そしてハード・コンフィギュレーション・ポート108aはコンフィギュレーション・バス112aを介してパイプライン回路80aのハード・コンフィギュレーション・ポート98aと結合されている。

【0081】

同様に、ファームウェア・メモリ52bは試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート104b、106b、108bを含み、それらが図4の試験、プログラミング、並びに、ハード・コンフィギュレーション・ポート104、106、108とそれぞれ類似している。試験ポート104bは試験バス63と結合され、プログラミング・ポート106bはプログラミング・バス110を介して通信インターフェース82のプログラミング・ポート94aと結合され、そしてハード・コンフィギュレーション・ポート108bはコンフィギュレーション・バス112bを介してパイプライン回路80bのハード・コンフィギュレーション・ポート98bと結合されている。

【0082】

図3、図5、並びに、図6で参照されるように、ピア-ベクトル・マシン40の動作（特にホストプロセッサ42、パイプライン回路80a及び80b、ファームウェア・メモリ52a及び52bの動作）は本発明の実施例に従って以下に議論されている。

【0083】

ピア-ベクトル・マシン40が先ず電源投入されると、ホストプロセッサ42は、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献3で議論されたようにそれ自体を初期化し、加速器44はそれ自体を部分的に初期化する。より詳細には、この部分的初期化中、パイプライン回路80a及び80bは、それぞれ、ファームウェア・メモリ52a及び52bの区分114a及び114bから初期コンフィギュレーション・ファームウェアをダウンロードする。それぞれの初期コンフィギュレーションにおいて、パイプライン回路80aは少なくとも通信インターフェース82及び試験回路（不図示）を含み、パイプライン回路80bは少なくとも試験回路（不図示）を含む。パイプライン回路80a及び80bがそれらの各初期コンフィギュレーションに構成された後、各パイプライン回路内の試験回路はパイプライン回路の各自試験を実行し（パイプライン回路80a及び80bの一方或は双方の試験回路もデータ・メモリ81を試験し得て）試験ポート96a及び96bのそれぞれと試験バス63とを介してホストプロセッサ42にそれら自己試験の結果を提供する。ファームウェア・メモリ52a及び52bも各自試験を実行し得て、図5と連携されて先に議論されたように、試験ポート104a及び104bのそれぞれと試験バス63とを介してホストプロセッサ42にその結果を提供する。

【0084】

次に、ホストプロセッサ42は加速器44の部分的初期化中に例外が生じたかを決定する。例えば、ホストプロセッサ42は試験バス63からの自己試験結果を分析して、パイプライン回路80a及び80b、データ・メモリ81、並びに、ファームウェア・メモリ

52a及び52bが適切に機能しているかを決定する。

【0085】

もし例外が生じたならば、ホストプロセッサ42はそれを所定の方法で取り扱う。例えば、もしホストプロセッサ42がパイプライン回路80aから自己試験結果を受信しなければ、初期コンフィギュレーション・ファームウェアがファームウェア・メモリ52aの区分114aに記憶されているかを試験バス63を介して検査し得る。もし初期コンフィギュレーション・ファームウェアが記憶されていなければ、ホストプロセッサ42はその初期コンフィギュレーション・ファームウェアを区分114aにロードして、パイプライン回路80aにそのファームウェアをダウンロードさせてから、自己試験の結果を分析し得る。この例はパイプライン回路50b及びファームウェア・メモリ52bにも適用される。例外のホストプロセッサの取り扱いは、先行して引用された「改善された計算アーキテクチャを有する計算マシン、関連システム、並びに、方法」と題された特許文献3に更に議論されている。

【0086】

もし例外が何等生じなかったならば、ホストプロセッサ42はパイプライン・ユニット124からプロファイル識別子を読み取り、続いて、図4と連携して先に議論されたように、加速器コンフィギュレーション・レジストリ70から或はファームウェア・メモリ52aの区分120からパイプライン・ユニットの対応するプロファイルを獲得する。

【0087】

次に、パイプライン・ユニット124（図4には1つのみ示されている）の全てからプロファイル識別子を読み取った後、ホストプロセッサ42は加速器44における全パイプライン・ユニットのマップを効果的に生成して、そのマップを例えばハンドラー・メモリ68に記憶する。

【0088】

次いで、ホストプロセッサ42はプロファイルからパイプライン回路80a及び80bの所望の動作コンフィギュレーションの識別子を抽出する。加速器44の初期化中に所望の動作コンフィギュレーションを抽出することは、そのプロファイルを初期化に先行して単に更新することによってパイプライン回路80a及び/或は80bの動作を変更させることを可能とする。

【0089】

次に、ホストプロセッサ42は所望の動作コンフィギュレーションを表すファームウェアがファームウェア・メモリ52a及び52bに記憶されているかを決定する。例えば、ホストプロセッサ42はプログラミング・バス110及び（パイプライン回路80aが初期コンフィギュレーションであるので、通信インターフェースが存在するので）通信インターフェース82を介してメモリ52aのメモリ区分118aからコンフィギュレーション記述を読み取ることができ、所望のファームウェアが区分116a₁、…、116a_nの何れかに記憶されていることを決定する。代替的には、ホストプロセッサ42は試験バス63及び試験ポート104aを介してメモリ52aから直にコンフィギュレーション記述を読み取り得る。この例もパイプライン回路50b及びファームウェア・メモリ52bに適用される。

【0090】

もし所望の動作コンフィギュレーションの一方或は双方を表すファームウェアがファームウェア・メモリ52a及び/或は52bに記憶されていなければ、ホストプロセッサ42は加速器コンフィギュレーション・レジストリ70からそのファームウェアを通信インターフェース82、プログラミング・ポート94、106、並びに、プログラミング・バス110を介して適切なファームウェア・メモリの区分116₁、…、116_nの内の1つにロードする。例えば、もしパイプライン回路80bの所望の動作コンフィギュレーションを表すファームウェアがメモリ52bに記憶されていなければ、ホストプロセッサ42はそのファームウェアをレジストリ70から、インターフェース82、プログラミング・ポート94、106b、並びに、プログラミング・バス110を介して区分116b₁、…、116b_nの内の1つにロードする。

6 b_iの内の1つにロードする。もしファームウェアがレジストリ70になれば、ホストプロセッサ42はそのファームウェアを外部ライブラリ（不図示）から検索し得るか、或は、例外指標を生成し得て、システムオペレータ（不図示）がそのファームウェアをレジストリ70にロードし得るように為す。

【0091】

次にホストプロセッサ42は、パイプライン回路80aに、ポート108a、コンフィギュレーション・バス112a、並びに、ポート98aを介してメモリ52aの対応する区分116a_i、-116a_jから所望のファームウェアをダウンロードするように命令し、パイプライン回路80bに、ポート108b、コンフィギュレーション・バス112b、並びに、ポート98bを介してメモリ52bの対応する区分116b_i、-116b_jから所望のファームウェアをダウンロードするように命令する。

10

【0092】

パイプライン回路80a及び80bが所望のファームウェアをダウンロードした後、それは所望の動作コンフィギュレーションにあって、データ処理を始める準備が為される。しかし、パイプライン回路80a及び80bがそれらの所望の動作コンフィギュレーションになった後でさえ、ホストプロセッサ42は新しいファームウェアを通信インターフェース82或は試験バス63を介して、図4と連携されて先に議論されたものと同様な方式でメモリ52a及び52bの区分116_i、-116_jにロードし得る。

【0093】

先行する議論は当業者が本発明を作製し使用することを可能とすべく提示されている。種々実施例への様々な変更は当業者には容易に明かであろうし、ここでの包括的な原則は本発明の精神及び範囲から逸脱することなしに他の実施例及び適用例に適用され得る。よって、本発明は図示された実施例に限定されることが意図されておらず、ここに開示された原理及び特徴と一貫した最も広い範囲と一致されるべきものである。

20

【図面の簡単な説明】

【0094】

【図1】図1は、従来の多数プロセッサ・アーキテクチャを有する計算マシンのブロック線図である。

【図2】図2は、従来のハードウェアに組み込まれたパイプラインのブロック線図である。

30

【図3】図3は、本発明の実施例に従ったピア-ベクトル・アーキテクチャを有する計算マシンのブロック線図である。

【図4】図4は、本発明の実施例に従った図3のパイプライン加速器におけるパイプライン・ユニットのブロック線図である。

【図5】図5は、本発明の実施例に従った図4のファームウェア・メモリの論理区画の線図である。

【図6】図6は、本発明の別の実施例に従った図3のパイプライン加速器のパイプライン・ユニットのブロック線図である。

【符号の説明】

【0095】

40

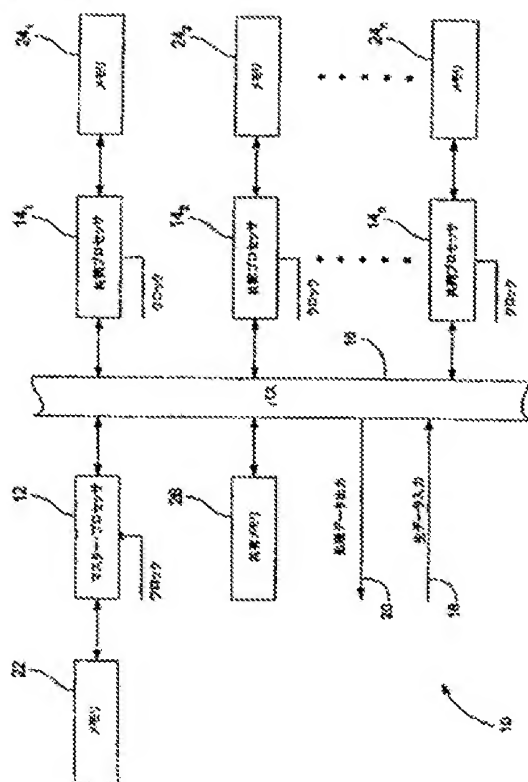
- 10 計算マシン
- 14 共同プロセッサ
- 40 ピア-ベクトル・マシン
- 42 ホストプロセッサ
- 44 パイプライン加速器
- 46 プロセッサ・メモリ
- 48 インターフェース・メモリ
- 50 パイプライン・バス
- 52 ファームウェア・メモリ
- 54 生データ入力ポート

50

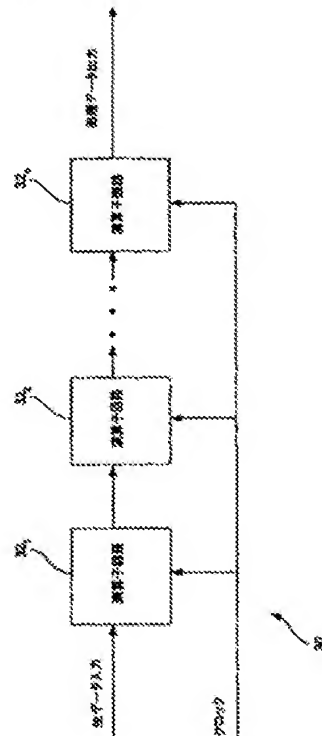
- 5 8 処理済みデータ出力ポート
- 6 1 ルータ
- 6 2 処理ユニット
- 6 6 処理ユニット・メモリ
- 6 8 ハンドラー・メモリ
- 7 0 加速器コンフィギュレーション・レジストリ
- 7 2 メッセージ・コンフィギュレーション・レジストリ
- 7 4 ハードウェアに組み込まれたパイプライン
- 7 8 パイプライン・ユニット
- 8 0 パイプライン回路
- 8 6 パイプライン・コントローラ
- 8 8 例外マネージャ
- 9 0 コンフィギュレーション・マネージャ
- 9 1 工業規格バス・インターフェース
- 9 3 通信バス

10

【図 1】



【図 2】



フロントページの続き

(31)優先権主張番号 10/684,053
 (32)優先日 平成15年10月19日(2003.10.9)
 (33)優先権主張国 米国(US)
 (31)優先権主張番号 10/684,057
 (32)優先日 平成15年10月9日(2003.10.9)
 (33)優先権主張国 米国(US)
 (31)優先権主張番号 10/684,102
 (32)優先日 平成15年10月9日(2003.10.9)
 (33)優先権主張国 米国(US)

(81)指定国 AP(BW, CH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), EA(AM, AZ, BY, KG, KZ, MD, RU, TJ, TW), EP(AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GR, GU, IE, IT, LI, MC, NL, PT, RO, SE, SI, SK, TR), OA(BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG), AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW

(74)代理人 100135585
 弁理士 西尾 務
 (72)発明者 ラーブ, ジョン, ダブリュ.
 アメリカ合衆国 バージニア州 20110 マナサッス, リバー クレスト ロード 9350
 (72)発明者 ジャクソン, ラリー
 アメリカ合衆国 バージニア州 20110 マナサッス, リバー クレスト ロード 9350
 (72)発明者 ジョーンズ, マーク
 アメリカ合衆国 バージニア州 20120 セントレビル, オークマー プレイス 15342
 (72)発明者 カーサロ, トロイ
 アメリカ合衆国 バージニア州 22701 カルベパー, ケストラル コート 1524
 フターム(参考) 58076 BBO6 EBO1
 58176 BBO6 EBO1